Lab J: Working with Images

Due April 7 9 pm

1 Introduction

In this lab we will do some work with images. More specifically, we will work to align colour channels of images; you will undoubtedly do related image manipulations should you ever take a computer vision class.

This assignment is inspired by one originally designed by Deva Ramanan and Alyosha Efros for students at CMU.

2 Programming Task

The RGB color model is a model in which the red, green and blue primary colors of light are added together in various ways to create different colors. The name of the model comes from the initials of the primary colors: red, green, and blue. You have been given the red, green, and blue channels of an image that were taken separately; your job in this homework is to assemble them in a reasonable way.

To load the various image channels, type the following command into a script or your console:

>> load image_channels.mat

This will load three matrices into your work space; each holds a different colour channel for a single image. The names of the variables you should have in your workspace after loading the file will be *redChannel, blueChannel* and *greenChannel*. Simply combining these channels in a 3-channel matrix to create a single image won't work, as the channels are not properly aligned! Figure 1 shows what will happen if you simply combine the channels without doing any additional work.

Your job is take the three colour channels and align them in order to produce a single, good looking image as output. This means you must write a function that find the "best possible displacement" between the various channels of colour.

The easiest way to find the "best possible displacement" is to exhaustively search over all possible displacements. For the purpose of this homework, you

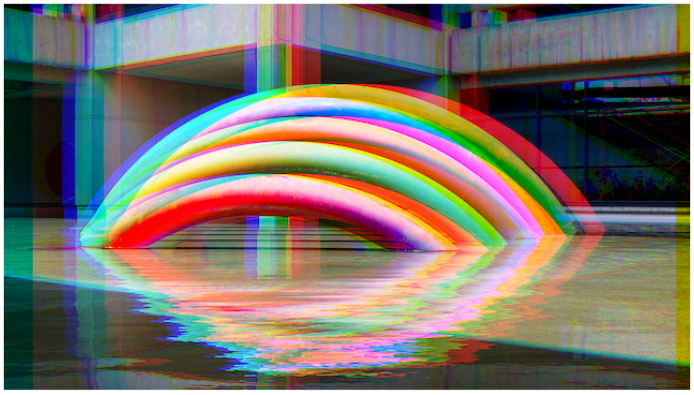

Figure 1: The three channels combined, without alignment!

can assume that the displacement between any two channels will be between -30 and 30 pixels and that the displacement will be along the **horizontal** axis of the image, not the vertical one. You can therefore score each possible horizontal displacement between -30 and 30 using a heuristic; then you will choose the best alignment by picking the alignment with the best score.

The scoring metric we will use is the sum of squared differences, which is much like the Euclidean distance metric that we have seen before. This metric compares the element-wise distance between the colour values that are held within matrices. More specifically, for two colour channels r and g of equal dimensions, the sum of squared differences can be defined as follows:

$$score(r, g) = \Sigma_{i=1}^{M} \Sigma_{j=1}^{N} (r_{i,j} - g_{i,j})^2$$

In the equation above, $r_{i,j}$ refers to the intensity of the pixel in the ith row and jth column of channel r while $g_{i,j}$ refers to the intensity of the corresponding pixel in channel g.

With this in mind, you are asked to implement a function with the following signature $combined = alignChannels(r, g, b)$ which:

1. accepts three image channels that are represented as matrices of intensity values, r, g and b.
2. returns a single image *combined* that combines the three channels, once they have been aligned.

This algorithm should:

1. start by identifying the channel (r, g or b) that contains the fewest pixels with an intensity of zero. Call this channel the *target* channel and align the other channels to match this one.
2. to align a channel c with the *target* channel you must search and score all possible displacements d (between -30 and 30 along the **horizontal** axis) for c. For each possible displacement d:
    - create a matrix m that is the identical to c but with pixels displaced by d. Look up the matlab function $circshift$ as it will help you make these displaced versions of c.
    - calculate $score(m, target)$ using the equation above. Remember that you can use element wise operations to speed up your calculations.
    - save every score you calculate for each displacement d.
3. once you have scored every possible displacement, locate the displacement associated with the **lowest** score you have generated. Then, displace the pixel intensities in c by this amount.

Figure 2: The three channels combined, with alignment!

4. finally, merge all of the channels once they have been aligned. Note that to combine three channels into a single image, you can use the matlab command $combined = cat(3, r, g, b);$.

When you are done, your aligned image should look something like the image in Figure 2.

3 What to Submit

Submit, via eClass:

1. Your **alignChannels.m** file

You will receive 1/2 mark for a working script and 1/2 a mark for a well commented, logical script. Your code should follow the function template presented in class; your function should therefore be commented and there should be output when we type 'help' for any function you have written into the console. Your code should pass all of the tests in the file 'labJtest.m' as well as any additional tests that we may write!!

HAVE FUN AND GOOD LUCK!